Importing and Exporting Items via Simple Archive Format

Item Importer and Exporter

DSpace has a set of command line tools for importing and exporting items in batches, using the DSpace Simple Archive Format. Apart from the offered functionality, these tools serve as a prime example for users who aim to implement their own item importer.

DSpace Simple Archive Format

The basic concept behind the DSpace's Simple Archive Format is to create an archive, which is directory full of items, with a subdirectory per item. Each item directory contains a file for the item's descriptive metadata, and the files that make up the item.

```
archive_directory/
    item_000/
        dublin_core.xml         -- qualified Dublin Core metadata for metadata fields belonging to the dc schema
        metadata_[prefix].xml   -- metadata in another schema, the prefix is the name of the schema as
registered with the metadata registry
        contents                -- text file containing one line per filename
        file_1.doc              -- files to be added as bitstreams to the item
        file_2.pdf
    item_001/
        dublin_core.xml
        contents
        file_1.png
        ...
```

The `dublin_core.xml` or `metadata_[prefix].xml` file has the following format, where each metadata element has it's own entry within a `<dcvalue>` tagset. There are currently three tag attributes available in the `<dcvalue>` tagset:

- `<element>` - the Dublin Core element
- `<qualifier>` - the element's qualifier
- `<language>`- (optional)ISO language code for element

```
<dublin_core>
    <dcvalue element="title" qualifier="none">A Tale of Two Cities</dcvalue>
    <dcvalue element="date" qualifier="issued">1990</dcvalue>
    <dcvalue element="title" qualifier="alternative" language="fr">J'aime les Printemps</dcvalue>
</dublin_core>
```

(Note the optional language tag attribute which notifies the system that the optional title is in French.)

Every metadata field used, must be registered via the metadata registry of the DSpace instance first, see Metadata and Bitstream Format Registries.

Recommended Metadata

It is recommended to minimally provide "dc.title" and, where applicable, "dc.date.issued". Obviously you can (and should) provide much more detailed metadata about the Item. For more information see: Metadata Recommendations.

The `contents` file simply enumerates, one file per line, the bitstream file names. See the following example:

```
file_1.doc
        file_2.pdf
        license
```

Please notice that the `license` is optional, and if you wish to have one included, you can place the file in the .../item_001/ directory, for example.

The bitstream name may optionally be followed by any of the following:

- `\tbundle:BUNDLENAME`
- `\tpermissions:PERMISSIONS`
- `\tdescription:DESCRIPTION`
- `\tprimary:true`

Where '\t' is the tab character.

'BUNDLENAME' is the name of the bundle to which the bitstream should be added. Without specifying the bundle, items will go into the default bundle, ORIGINAL.

'PERMISSIONS' is text with the following format: `-[r|w] 'group name'`

'DESCRIPTION' is text of the files description.

Primary is used to specify the primary bitstream.

Configuring `metadata_[prefix].xml` for Different Schema

It is possible to use other Schema such as EAD, VRA Core, etc. Make sure you have defined the new scheme in the DSpace Metada Schema Registry.

1. Create a separate file for the other schema named `metadata_[prefix].xml`, where the `[prefix]` is replaced with the schema's prefix.
2. Inside the xml file use the dame Dublin Core *syntax*, but on the `<dublin_core>` element include the attribute `schema=[prefix]`.
3. Here is an example for ETD metadata, which would be in the file `metadata_etd.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<dublin_core schema="etd">
     <dcvalue element="degree" qualifier="department">Computer Science</dcvalue>
     <dcvalue element="degree" qualifier="level">Masters</dcvalue>
     <dcvalue element="degree" qualifier="grantor">Texas A & M</dcvalue>
</dublin_core>
```

Importing Items

Before running the item importer over items previously exported from a DSpace instance, please first refer to Transferring Items Between DSpace Instances.

Command used:	`[dspace]/bin/dspace import`
Java class:	`org.dspace.app.itemimport.ItemImport`
Arguments short and (long) forms:	Description
`-a or --add`	Add items to DSpace ‡
`-r or --replace`	Replace items listed in mapfile ‡
`-d or --delete`	Delete items listed in mapfile ‡
`-s or --source`	Source of the items (directory)
`-c or --collection`	Destination Collection by their Handle or database ID
`-m or --mapfile`	Where the mapfile for items can be found (name and directory)
`-e or --eperson`	Email of eperson doing the importing
`-w or --workflow`	Send submission through collection's workflow
`-n or --notify`	Kicks off the email alerting of the item(s) has(have) been imported
`-t or --test`	Test run, do not actually import items
`-p or --template`	Apply the collection template
`-R or --resume`	Resume a failed import (Used on Add only)
`-h or --help`	Command help
`-z or --zip`	Name of zipfile

‡ These are mutually exclusive.

The item importer is able to batch import unlimited numbers of items for a particular collection using a very simple CLI command and 'arguments'

Adding Items to a Collection from a directory

To add items to a collection, you gather the following information:

- eperson
- Collection ID (either Handle (e.g. 123456789/14) or Database ID (e.g. 2)
- Source directory where the items reside
- Mapfile. Since you don't have one, you need to determine where it will be (e.g. /Import/Col_14/mapfile)
  At the command line:

```
[dspace]/bin/dspace import --add --eperson=joe@user.com --collection=CollectionID --source=items_dir --
mapfile=mapfile
```

or by using the short form:

```
[dspace]/bin/dspace import -a -e joe@user.com -c CollectionID -s items_dir -m mapfile
```

The above command would cycle through the archive directory's items, import them, and then generate a map file which stores the mapping of item directories to item handles. **SAVE THIS MAP FILE.** Using the map file you can use it for replacing or deleting (unimporting) the file.

Testing. You can add `--test` (or `-t`) to the command to simulate the entire import process without actually doing the import. This is extremely useful for verifying your import files before doing the actual import.

Adding Items to a Collection from a zipfile

To add items to a collection, you gather the following information:

- eperson
- Collection ID (either Handle (e.g. 123456789/14) or Database ID (e.g. 2)
- Source directory where your zipfile containing the items resides
- Zipfile
- Mapfile. Since you don't have one, you need to determine where it will be (e.g. /Import/Col_14/mapfile)
  At the command line:

```
[dspace]/bin/dspace import --add --eperson=joe@user.com --collection=CollectionID --source=items_dir --
zip=filename.zip --mapfile=mapfile
```

or by using the short form:

```
[dspace]/bin/dspace import -a -e joe@user.com -c CollectionID -s items_dir -z filename.zip -m mapfile
```

The above command would unpack the zipfile, cycle through the archive directory's items, import them, and then generate a map file which stores the mapping of item directories to item handles. **SAVE THIS MAP FILE.** Using the map file you can use it for replacing or deleting (unimporting) the file.

Testing. You can add `--test` (or `-t`) to the command to simulate the entire import process without actually doing the import. This is extremely useful for verifying your import files before doing the actual import.

Replacing Items in Collection

Replacing existing items is relatively easy. Remember that mapfile you saved above? Now you will use it. The command (in short form):

```
[dspace]/bin/dspace import -r -e joe@user.com -c collectionID -s items_dir -m mapfile
```

Long form:

```
[dspace]/bin/dspace import --replace --eperson=joe@user.com --collection=collectionID --source=items_dire --
mapfile=mapfile
```

Deleting or Unimporting Items in a Collection

You are able to unimport or delete items provided you have the mapfile. Remember that mapfile you saved above? The command is (in short form):

```
[dspace]/bin/dspace import -e joe@user.com -d -m mapfile
```

In long form:

```
[dspace]/bin/dspace import --eperson=joe@user.com --delete --mapfile mapfile
```

Other Options

- **Workflow**. The importer usually bypasses any workflow assigned to a collection. But add the `--workflow` (`-w`) argument will route the imported items through the workflow system.
- **Templates**. If you have templates that have constant data and you wish to apply that data during batch importing, add the `--template` (`-p`) argument.
- **Resume**. If, during importing, you have an error and the import is aborted, you can use the `--resume` (`-R`) flag that you can try to resume the import where you left off after you fix the error.

Exporting Items

The item exporter can export a single item or a collection of items, and creates a DSpace simple archive according to the aforementioned format for each item to be exported. The items are exported in a sequential order in which they are retrieved from the database. As a consequence, the sequence numbers of the item subdirectories (item_000, item_001) are not related to DSpace handle or item id's.

Command used:	`[dspace]/bin/dspace export`
Java class:	*org.dspace.app.itemexport.ItemExport*
Arguments short and (long) forms:	Description
`-t` or `--type`	Type of export. *COLLECTION* will inform the program you want the whole collection. *ITEM* will be only the specific item. (You will actually key in the keywords in all caps. See examples below.)
`-i` or `--id`	The ID or Handle of the Collection or Item to export.
`-d` or `--dest`	The destination of where you want the file of items to be placed. You place the path if necessary.
`-n` or `--number`	Sequence number to begin export the items with. Whatever number you give, this will be the name of the first directory created for your export. The layout of the export is the same as you would set your layout for an Import.
`-m` or `--migrate`	Export the item/collection for migration. This will remove the handle and metadata that will be re-created in the new instance of DSpace.
`-h` or `--help`	Brief Help.

Exporting a Collection

To export a collection's items you type at the CLI:

```
[dspace]/bin/dspace export --type=COLLECTION --id=collID --dest=dest_dir --number=seq_num
```

Short form:

```
[dspace]/bin/dspace export -t COLLECTION -i [CollID or Handle] -d /path/to/destination -n Some_number
```

Exporting a Single Item

The keyword *COLLECTION* means that you intend to export an entire collection. The ID can either be the database ID or the handle. The exporter will begin numbering the simple archives with the sequence number that you supply. To export a single item use the keyword *ITEM* and give the item ID as an argument:

```
[dspace]/bin/dspace export --type=ITEM --id=itemID --dest=dest_dir --number=seq_num
```

Short form:

```
[dspace]/bin/dspace export -t ITEM -i [itemID or Handle] -d /path/to/destination -n some_number
```

Each exported item will have an additional file in its directory, named 'handle'. This will contain the handle that was assigned to the item, and this file will be read by the importer so that items exported and then imported to another machine will retain the item's original handle.

The `-m` Argument

Using the `-m` argument will export the item/collection and also perform the migration step. It will perform the same process that the next section Exchanging Content Between Repositories performs. We recommend that section to be read in conjunction with this flag being used.